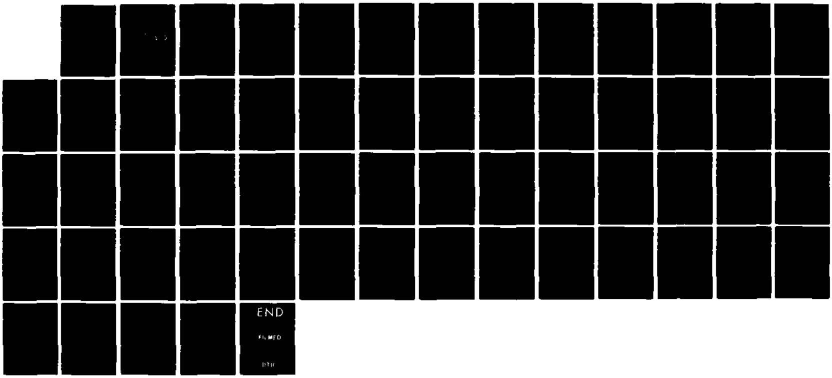
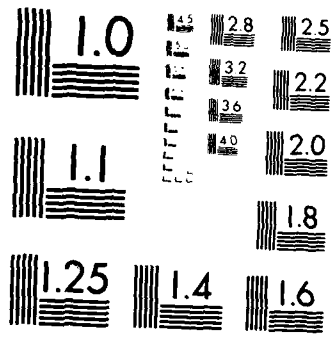
AD-A152 135 NUMERICAL OPTIMIZATION ALGORITHM FOR ENGINEERING PROBLEMS USING MICROCOMPUTER(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA D S KIM SEP 84 1/1

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

2

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

NUMERICAL OPTIMIZATION ALGORITHM
FOR ENGINEERING PROBLEMS
USING MICROCOMPUTER

by

Dong Soo, Kim

September 1984

Thesis Advisor: G. N. Vanderplaats

Approved for public release; distribution unlimited.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A152135	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Numerical Optimization Algorithm for Engineering Problems Using Microcomputer		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; September 1984
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Dong Soo, Kim		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		12. REPORT DATE September 1984
		13. NUMBER OF PAGES 57
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microcomputer
Feasible Direction Method

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A general purpose computer program is developed to perform nonlinear constrained optimization of engineering design problems. The program is developed especially for use on microcomputers and is called Microcomputer Software for Constrained Optimization Problems (MSCOP). It will accept a nonlinear objective function and up to 50 inequality constraint functions and up to 20 bounded design variables.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

MSCOP employs the method of feasible directions. Although developed for microcomputers, for speed of development, the MSCOP was implemented on an IBM 3033 using standard basic language, Waterloo BASIC Version 2.0. It is directly transportable to a variety of microcomputers.

Typical applications of MSCOP program are in the design of machine components and simple beam and truss structures. Solutions to three sample problems are given.

Approved for public release; distribution unlimited.

Numerical Optimization Algorithm for Engineering Problems Using Micro-computer

by

Dong Soo, Kim
Major, Republic of Korea Army
B.S., Korea Military Academy, 1976
B.E., Seoul National University, 1980

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1984

Author: Kim Dong Soo
Dong Soo, Kim

Approved by: Garret N. Vanderplaats
Garret N. Vanderplaats, Thesis Advisor

R. Kevin Wood
R. Kevin Wood, Second Reader

R. T. Howard, Acting
Alan R. Washburn, Chairman,
Department of Operations Research

Kneale T. Marshall
Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

A general purpose computer program is developed to perform nonlinear constrained optimization of engineering design problems. The program is developed especially for use on microcomputers and is called Microcomputer Software for Constrained Optimization Problems (MSCOP). It will accept a nonlinear objective function and up to 50 inequality constraint functions and up to 20 bounded design variables.

MSCOP employs the method of feasible directions. Although developed for microcomputers, for speed of development, the MSCOP was implemented on an IBM 3033 using standard basic language, Waterloo BASIC Version 2.0. It is directly transportable to a variety of microcomputers.

Typical applications of MSCOP program are in the design of machine components and simple beam and truss structures. Solutions to three sample problems are given.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENT

I am very grateful to Prefessor Garret N. Vanderplaats whose expert advise, technical support and guidance resulted in my much understanding of engineering optimization.

I also wish to express my sincere appreciation to Professor R. Kevin Wood for his critical review and comments during the preparation of this thesis.

The author extends a special thanks to Dr. Noriyaki Yoshida for his assistance, his time and his patience during the course of work.

Finally, to my wife, Jong Soon, whose patience and support was instrumental in the completion of this work.

I. INTRODUCTION

A. PURPOSE

This thesis describes the development of a microcomputer oriented program called MSCOP (Microcomputer Software for Constrained Optimization Problems) for constrained optimization of engineering design problems. Problems which can be solved by the MSCOP are nonlinear programming problems arising in several areas of machine and structural design, such as the minimum weight design of structures subject to stress and displacement constraints [Ref. 1].

In recent years, several powerful general purpose optimization programs have become available for engineering design problems, e.g., COPES/CONMIN [Ref. 2], and ADS-1 [Ref. 3]. These programs can handle a wide range of design problems and contain a variety of solution techniques. Also, several programs are available that include optimization in an integrated analysis / design code, e.g., ACCESS, ASOP, EAL, PARS, SAVES, SPAR, STARS and TSO [Ref. 4]. All of the above optimization programs are written in FORTRAN, and are built for use on a mainframe computer. Their use can be cumbersome, especially for the occasional user. Since many engineers are now using microcomputers, there is a need to develop an optimization program contained in a microcomputer software package for use on microcomputers. This thesis fills that need by developing a compact program written in a standard BASIC language suitable for a wide range of microcomputers.

B. IMPLEMENTATION

The nature of an optimization program depends on the computer and programming method available. The MSCOP software is designed for use on a microcomputer. However, for the speed of development and testing, MSCOP was developed on the IBM 3033 computer at the W. R. Church Computer Center in Naval Postgraduate School, and was written in WBASIC (Waterloo Basic) Version 2.0.

To make sure that the program is easily portable to a microcomputer, only standard BASIC commands and functions are used. For example, FOR I = 1 TO NDB ... NEXT I, GOSUB etc., were used. The commands and functions not available in all variations of BASIC are avoided, for example, TRN(A), MAT(A), etc.

MSCOP provides design engineers with a convenient tool for optimization of engineering design problems with up to 20 bounded design variables and as many as 50 inequality constraints.

C. GENERAL OPTIMIZATION MODEL

The general optimization problem to be solved is of the form : Find the set of design variables $\underline{X}$ that will

$$\text{Minimize} \quad F(\underline{X}) \tag{1.1}$$

$$\text{Subject to} \quad G_j(\underline{X}) \leq 0 \qquad j = 1,\ldots, m \tag{1.2}$$

$$X_i^l \leq X_i \leq X_i^u \qquad i = 1,\ldots, n \tag{1.3}$$

where X is referred to as the vector of design variables. $F(\underline{X})$ is the objective function which is to be minimized. $G(\underline{X})$ are inequality constraint functions, and X_i^l and X_i^u are lower and upper bounds, respectively, on the design

variables. Although these bounds or "side constraints" could be included in the inequality constraint set given by Eq(1.2), it is convenient to treat them separately because of their special structure. The objective function and constraint functions may be nonlinear, explicit or implicit in X. However, they must be continuous and should have continuous first derivatives.

In general engineering optimization problems, the objective to be minimized is usually the weight or volume of a structure being designed while the constraints gives limits on compressive stress, tensile stress, Euler buckling, displacement, frequencies (eigenvalues), etc. [Ref. 5 : p.264]. Equality constraints are not included because their inclusion complicates the solution techniques and because in engineering situations, equality constraints are rare.

Most optimization algorithms require that an initial value of design variables X^0 be specified. Beginning from these starting values, the design is iteratively improved. The iterative procedure is given by

$$\underline{X}^{q+1} = \underline{X}^{q} + a^* \underline{S}^{q} \tag{1.4}$$

where q is the iteration number, S is a search direction vector in the design space, and a* is a scalar parameter which defines the amount of change in $\underline{X}$. At iteration q, it is desirable to determine a direction $\underline{S}$ which will reduce the objective function (usable direction) without violating the constraints (feasible direction). After determining the search direction, the design variables, $\underline{X}$, are updated by Eq (1.4) so that the minimum objective value is found in this direction. [Ref. 6].

Thus, it is seen that nonlinear optimization algorithms for the general optimization problem based on Eq(1.4) can be separated into two parts, determination of search direction and determination of scalar parameter a*.

D. ORGANIZATION OF THIS THESIS

This chapter has stated the purpose of the thesis and has put the general concept of engineering optimization into a preliminary perspective. Chapter 2 will describe the essential aspects of the optimization algorithm used in MSCOP such as finding a search direction, the one-dimensional search and convergence criteria. Chapter 3 describes program usage. In chapter 4, there are three examples which are solved by the MSCOP. Summary and conclusions are given in chapter 5. The program is listed in the appendix.

II. OPTIMIZATION ALGORITHM

A. INTRODUCTION

There are many optimization algorithms for constrained nonlinear problems such as generalized reduced gradient method, feasible direction method, penalty function methods, Augmented Lagrangian multiplier method, and sequential linear programming. The feasible direction method is chosen for development in this thesis for three main reasons. First it progresses rapidly to a near optimum design. Second it only requires gradients of objective and constraint functions that are active at any given point in the optimization process [Ref. 7]. Third, because it maintains a feasible design, engineer cannot fail to meet safety requirements as defined by the contraints. However, the method does have several disadvantages in that it is prone to "zig-zag" between constraint boundaries and that it is usually does not achieve a precise optimum. This method solves the nonlinear programming problem by moving from a feasible point (can be initially infeasible) to another feasible point with an improved value of the objective value.

The following strategy is typical of feasible direction method : Assuming that an initial feasible point X^0 is known, first find a usable-feasible direction S. The algorithm for this is similar to linear programming and complementary pivoting algorithms. Having found the search direction, a move is made in this direction to update the X vector according to Eq(1.4). The scalar a* is found by a one-dimensional search to reduce the objective function as much as possible subject to constraints. That is MIN

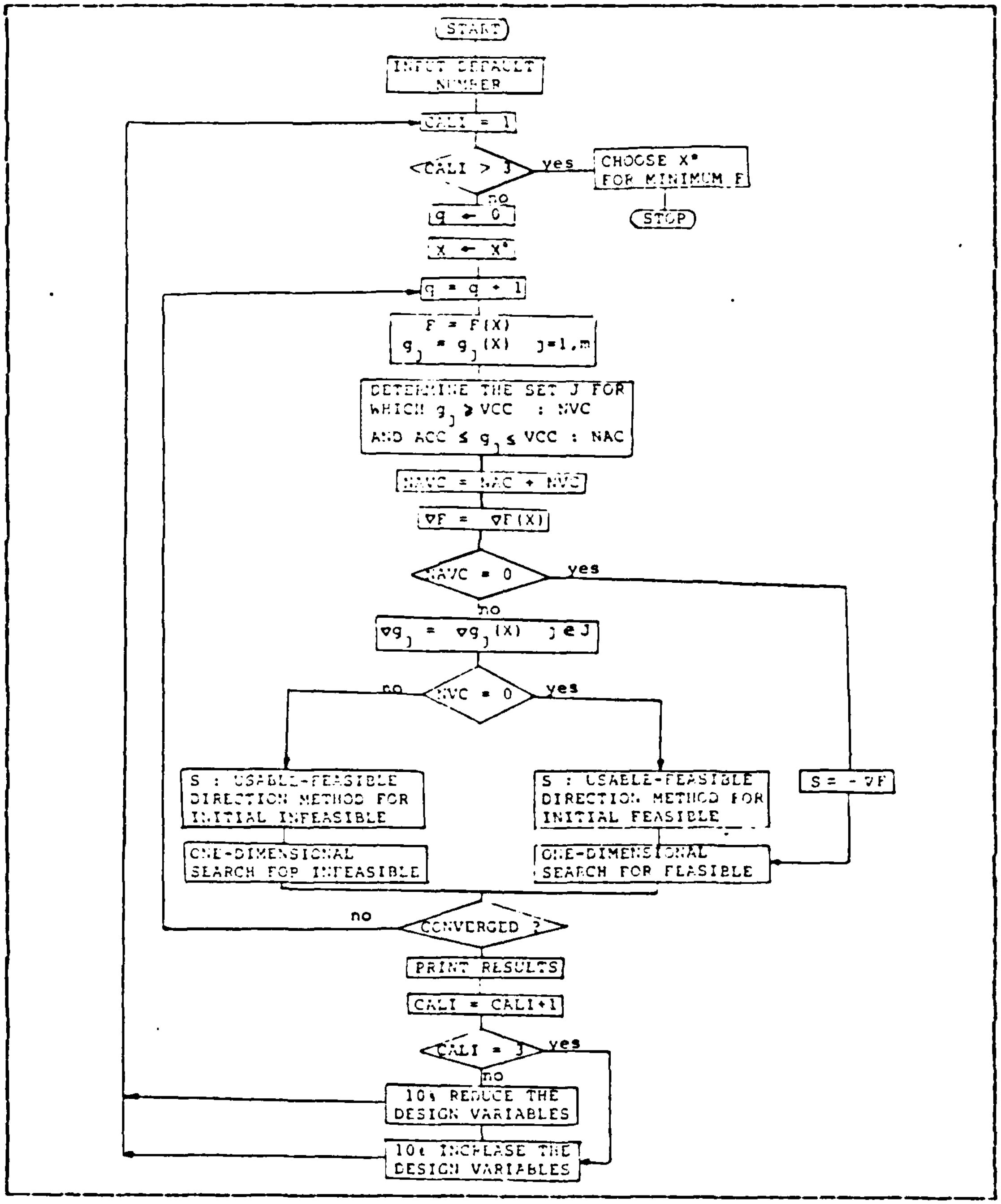


Figure 2.1 Algorithm for the Feasible Direction Method.

$F(\underline{X}+a*\underline{S})$ subject to $G(\underline{X}+a*\underline{S}) \leq 0$. It is assumed that the initial design $\underline{X}^0$ is feasible, but if it is not, a search

direction is found which will direct the design to the feasible region. After updating the X^0 vector, the convergence test must be performed in the iterative algorithm. A convergence criteria used in this is implementation are described in section D. The general algorithm used in MSCOP is given in Figure 2.1

B. SEARCH DIRECTION

In the feasible direction algorithm, a usable - feasible search direction S is found which will reduce the objective function without violating any constraints for some finite move. It is assumed that at any point in the design space (at any $\underline{X}$) the value of the objective and constraint functions as well as the gradients of these functions with respect to the design variables can be calculated. Since these gradients cannot usually be calculated analytically, the finite difference method Eq(2.1) is used in MSCOP.

$$\frac{\partial F(\underline{X})}{\partial X_i} = \frac{F(\underline{X}+ \varepsilon e_i) - F(\underline{X})}{\varepsilon} \tag{2.1}$$

where e_i is the ith unit vector

ε is a small scalar.

In MSCOP, ε is 0.1% of the ith design variable

In the feasible direction algorithm, there are usually one or more "active" constraints. A constraint $G(\underline{X}) \leq 0$ is "active" at $\underline{X}$ if $g(\underline{X}) \approx 0$. As shown in Figure 2.1, if no constraints are active the standard steepest descent direction $\underline{S} = -\underline{\nabla}F$ is used.

1. Usable-Feasible Direction

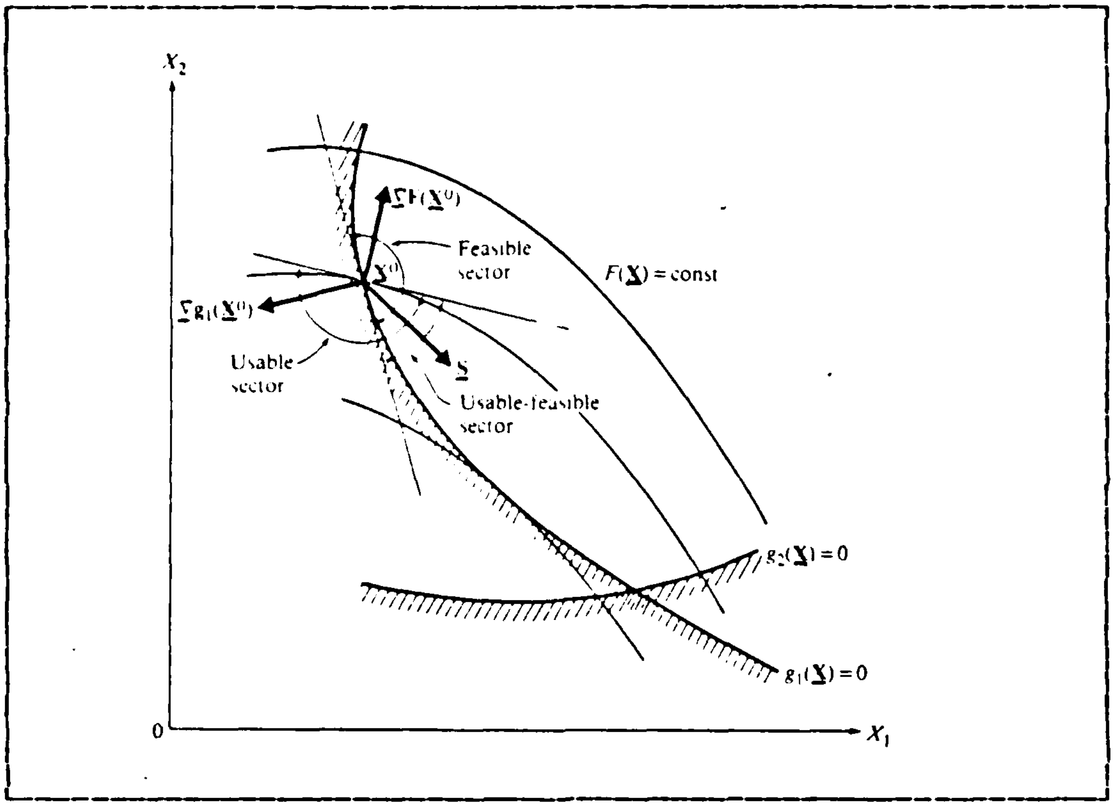


Figure 2.2 Usable-Feasible Direction.

Assume there are NAC active constraints at $\underline{X}$. The direction $\underline{S}$ is "usable" if it reduces the objective function, i.e.,

$$\nabla F \cdot S < 0 \tag{2.2}$$

Similarly the direction is feasible if for a small movement in this direction, no constraint will be violated, i.e.,

$$\nabla G \cdot S < 0 \tag{2.3}$$

This is shown geometrically in Figure 2.2

2. Active Constraints

It is necessary to determine if a constraint is active or violated in the feasible direction algorithm. A constraint $G(\underline{X}) \leq 0$ is "active" at $\underline{X}^0$ if $G(\underline{X}^0) \approx 0$. In order to avoid the zigzagging effect between one or more constraint boundaries, a tolerance band about zero is used for determining whether or not a constraint is active. From the engineering point of view, a constraint $G(\underline{X}) \leq 0$ is active near the boundary $G(\underline{X}) = 0$ whenever $ACC \leq G(\underline{X}) \leq VCC$. ACC is the active constraint criterion and VCC is the violated constraint criterion in MSCOP. Assuming the feasible constraints are normalized so that $G(\underline{X})$ ranges between -1 and 0 for reasonable values of $\underline{X}$, the constraint $G(X) \leq 0$ is considered active if $G(\underline{X}) \geq -0.1$. The constraint is considered to be violated if $G(\underline{X}) > 0.004$. This is an algorithmic trick which improves efficiency and reliability of the algorithm. However, since in the one - dimensional search, all interpolations for constraint $G(\underline{X})$ are done for zeros of a linear or quadratic approximation to $G(\underline{X})$ in order to find a*, at the optimum the value of active constraints are very near zero, but may be as large as 0.004 [Ref. 6]. From an engineering point of view, a 0.4 % constraint violation is considered to be acceptable.

3. Suboptimization Problem and Push-Off Factors

Zoutendijk [Ref. 8] has shown that a usable - feasible direction S may be found as follows :

$$\text{Maximize} \quad \beta \tag{2.4}$$

Subject to ;

$$\underline{\nabla}F(\underline{X})\cdot\underline{S} + \beta \leq 0 \tag{2.5}$$

$$\nabla G(X) \cdot S + \theta_j \beta \leq 0 \qquad j \in J \tag{2.6}$$

$$S \text{ bounded} \tag{2.7}$$

Where scalar β is a measure of the satisfaction of the usability and feasibility requirements. The scalar θ_j in Eq (2.6) is referred to as the "push-off" factor which effectively pushes the search direction away from the active

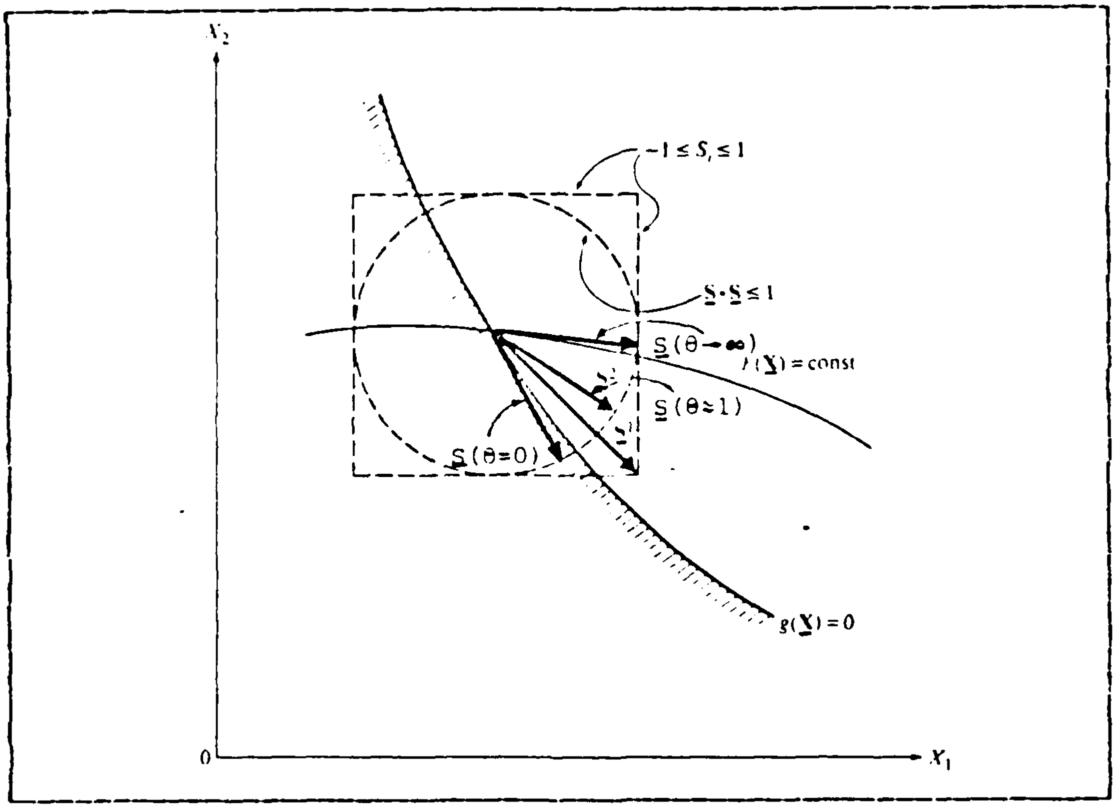


Figure 2.3 Push-Off Factor and Bounding of the S-Vector.

constraints. In Eq (2.6), if the push-off factor is zero, the search direction is tangent to the active constraints, and if it is infinite, then the search direction is tangent to the objective function. It has been found that a

push-off factor is defined as follows gives good results [Ref. 5: p.167] :

$$\theta_j = \left[1 - \frac{G_j(X)}{ACC} \right]^2 \theta_o \qquad (2.8)$$

where $\theta_o = 1$.

To avoid an unbounded solution when seeking a usable - feasible direction it is necessary to impose bounds on the search direction $\underline{S}$. One method of imposing bounds on search direction is to impose bounds on the components of S-vector of form :

$$-1 \leq S_i \leq 1 \qquad (2.9)$$

This choice of bounding the S-vector actually biases the search direction. This is undesirable since we wish to use the push-off factors as our means of controlling the search direction. A method which avoids this bias in search direction is the circle as shown Figure 2.3 . The norm here is

$$\underline{S} \cdot \underline{S} \leq 1 \qquad (2.9.1)$$

4. <u>Simple Simplex-like Method for Search Direction</u>

Vanderplaats [Ref. 5: pp. 168-169] provides the matrix formulation which solves the above sub-optimization problem by using the Zoutendijk method.

Maximize $\underline{P} \cdot \underline{y}$ (2.10)

Subject to ;

$$\underline{\underline{A}} \cdot \underline{y} \leq 0 \qquad (2.11)$$

$$\underline{y}\cdot\underline{y} \leq 1 \qquad (2.12)$$

Where

$$\underline{y} = \begin{bmatrix} \underline{S}_1 \\ \underline{S}_2 \\ \vdots \\ \underline{S}_n \\ \beta \end{bmatrix} \qquad \underline{P} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \qquad (2.13)$$

$$\underline{\underline{A}} = \begin{bmatrix} \underline{\nabla}^T G_1(X), & \theta_1 \\ \underline{\nabla}^T G_2(X), & \theta_2 \\ \vdots & \vdots \\ \underline{\nabla}^T G_j(X), & \theta_j \\ \underline{\nabla}^T F\ (X), & 1 \end{bmatrix} \qquad (2.14)$$

and where j is the number of active constraints (NAC)

When the solution to Eq(2.10) through (2.12) is found, S may be normalized to some value other than unity, but the form of the normalization is the same. A solution to the above problem may be obtained by solving the following system derived from the Kuhn-Tucker conditions for that problem :

$$\begin{bmatrix} B & I \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \underline{C} \qquad (2.15)$$

$$u_i \geq 0 \qquad v_i \geq 0 \qquad \underline{u}\cdot\underline{v} = 0 \qquad (2.16)$$

Where

$$\underline{\underline{B}} = -\underline{\underline{A}} \cdot \underline{\underline{A}}^T \tag{2.17}$$

$$\underline{\underline{I}} = \text{Identity matrix} \tag{2.18}$$

$$\underline{c} = -\underline{\underline{A}} \cdot \underline{P} \tag{2.19}$$

Above system can be solved using a complimentary pivot algorithm. Choose an initial basic solution to Eq(2.15) is to be

$$\underline{v} = \underline{c}, \qquad \underline{u} = 0 \tag{2.20}$$

where $\underline{v}$ is the set of basic variables and $\underline{u}$ is the set of nonbasic variables. If all $v_i > 0$, Eq(2.16) is also satisfied and problem is solved. If some $v_i < 0$, the solution procedure is as follows :

Let B_{ii} be the diagonal element of the i-th nonbasic variable.

1. Given the condition that some c is less then zero, we find max (c_i/B_{ii}) which is the incoming row to the basis.
2. The incoming column is changed to a basic column, the tableau is updated by a standard simplex pivot on B_{ii} .
3. Until all $c_i > 0$, repeat steps 1. and 2.
4. When all $c_i > 0$, the iteration is complete. The value of u is now the desired solution.
5. By using $\underline{y} = \underline{p} - \underline{\underline{A}}^T \cdot \underline{u}$, we get the usable-feasible search direction S which is first NDV components of y.

5. Initially Infeasible Designs

The method of feasible directions assumes that we begin with a feasible design and feasibility is maintained throughout the optimization process. If the initial design

is infeasible, then a search direction pointing toward the feasible region can be found by a simple modification to direction finding problem.

A design situation can exist in which the violated constraints are strongly dependent on part of the design variables, while the objective function is primarily dependent on the other design variables. This suggests a method for finding a search direction which will simultaneously minimize the objective while overcoming the constraint violations. These considerations lead to the following statement of the direction finding problem [Ref. 5 : pp. 171-172] :

$$\text{Maximize} \qquad -\underline{\nabla}F(\underline{X})\cdot\underline{S} + \Phi\beta \tag{2.21}$$

Subject to ;

$$\underline{\nabla}G(\underline{X})\cdot\underline{S} + \theta_j\beta \leq 0 \qquad j \in J \tag{2.22}$$

$$\underline{S}\cdot\underline{S} \leq 1 \tag{2.23}$$

where J is the set of active and violated constraints, and where the scalar Φ in Eq(2.21) is a weighting factor determining the relative importance of the objective and the constraints. Usually a value of $\Phi > 10000$ will ensure that the resulting S-vector will point toward the feasible region. Incorporating Eq(2.21) and Eq(2.22) into the direction finding algorithm requires only that we modify the p-vector given in Eq(2.24) and the A-matrix of Eq(2.25).

$$P = \begin{bmatrix} -\nabla F(\underline{X}) \\ \Phi \end{bmatrix} \tag{2.24}$$

$$\underline{\underline{A}} = \begin{bmatrix} \underline{\nabla}^T G_1(X), & \theta_1 \\ \underline{\nabla}^T G_2(X), & \theta_2 \\ \vdots & \vdots \\ \underline{\nabla}^T G_j(X), & \theta_j \end{bmatrix} \tag{2.25}$$

$$\theta_j \leq 50 \tag{2.26}$$

We use the simple simplex-like method to find the search direction toward the feasible region.

C. ONE-DIMENSIONAL SEARCH

1. No Violated Constraints

If no constraints are violated, we find the largest a* in Eq(1.4) from all possible values that will minimize the objective on S without violating any constraints, active or inactive.

The procedure in MSCOP is as follows :

1. Let a0, a1, a2, a3 be the scalar in Eq(1.4) corresponding to points $\underline{X0}$, $\underline{X1}$, $\underline{X2}$, $\underline{X3}$, $\underline{X4}$.
2. a0 = 0 at given point $\underline{X0}$.
3. In order to get a1, we can calculate the a1 to reduce the objective by at most 10% or to change each of the design variable $\underline{X}$ by at most 10%.
4. Update the design variables to $\underline{X1}$ using Eq(1.4).
5. Evaluate the objective for $\underline{X1}$, and check the feasibility. If one or more constraints is violated, then a1 is reduced to a1/2, and we go to step 4.
6. In order to estimate a2, we can use the quadratic approximation with 2 points $\underline{X}$, $\underline{X1}$ and the $\underline{\nabla}F$.

7. Update the design variables to $\underline{X}2$ by Eq(1.4) and check the side constraints.
8. Evaluate the objective and constraints.
9. Now having 3 a's, and values of objectives and constraints for design variables $\underline{X}0$, $\underline{X}1$, $\underline{X}2$ are known, so by using 3-point quadratic approximation, a value of a3 is found.
10. Update the new optimal point in search direction by Eq(1.4).
11. Evaluate the objective and constraints.
12. Now choose last 3 values, a1, a2, a3 and find a new a3 using 3-points Quadratic approximation
13. Choose the a* among the 5 points which corresponds to the minimum objective function value with no-violated constraints.

2. One or More Constraints Violated

If one or more constraints are initially violated, a modified usable-feasible direction is found. It is then necessary to find the scalar a* in Eq(1.4) which will minimize the maximum constraint violation, using the most violated constraint j, a good initial estimate for a* is

$$a^* = \frac{-G_j(\underline{X})}{\underline{\nabla} G_j(\underline{X}) \cdot \underline{S}} \tag{2.27}$$

Since the gradients of the violated constraints are known, the scalar which is required to obtain a feasible design with respect to violated constraint in the search direction, is given to a first approximation by Eq(2.27).

The more detail procedure in MSCOP is as follow ;

1. Choose the most violated constraint j.
2. Calculate a* for violated constraint j using Eq(2.27).

3. Update the design variables for a* and check the side constraints.
4. If one or more violated constraints still exist, then calculate the derivative of objective, violated and active constraints and find a new search direction and then go to step 1. Otherwise proceed with the optimization in the normal fashion.

D. CONVERGENCE CRITERIA

A desired property of an algorithm for solving a nonlinear problem is that it should generate a sequence of points converging to a global optimal point. In many cases, however, we may have to be satisfied with less faverable outcomes. In fact, as a result of non-convexity, problem size, and other difficulties, we may stop the iterative procedure if a point belongs to a described set, which is defined in MSCOP as follows ;

1. $Q_1 = \{ \underline{X} \mid |\underline{X}^o - \underline{X}| < \varepsilon_x \cdot |\underline{X}^o| \}$

2. $Q_2 = \{ \underline{X} \mid |F(\underline{X}^o) - F(\underline{X})| < \varepsilon_f \cdot |F(\underline{X}^o)| \}$

In MSCOP, the algorithm is terminated if a point $\underline{X}$ is reached such that $\underline{X} \in Q_1 \cap Q_2$. ε_x is 0.001 and ε_f is approximatly 0.001. Since in engineering design problems it is not necessary to find solutions with more than three significant digits.

III. MSCOP USAGE

A. INTRODUCTION

Since this MSCOP is written in WATERLOO BASIC Version 2.0, it is very convenient to use. The user must first formulate the design problem with the classical machine design criteria. Given the formulation of the design problem as a nonlinear program, the user then enters the problem as a part of a BASIC program. The user defines the objective function and constraint functions using BASIC statements. Other parameters are input as data : the number of design variables NDV, the number of inequality constraints NIQC, variable bounds an initial design value and a print control number.

B. PROBLEM FORMULATION

Generally, the experienced design engineer will be able to choose the appropriate objective for optimization depending on the requirements of the particular application. The physical phenomena of significance should first be summarized for the device to be designed. The appropriate objective can then be selected and constraints can be imposed on the remaining phenomena to assure an acceptable design from all standpoints. However, the initial formulation for the optimization problem should not be more complicated then necessary and this often requires the making of some simplifying assumptions. [Ref. 9].

After completing the formulation of the design problem, the design engineer should be able to answer the following questions :

1. What are the design variables ?

2. What is the objective function ?

3. What are the inequality constraints ?

4. What are the bounds on the variables ?

The engineer is then ready to input the program to the MSCOP. However, additional study and preparation of the problem may be useful. In particular, redundant constraints should be avoided if possible. MSCOP will operate with redundant constraints but it will operate faster without them. Selection of an initial design point from which to start this program is important, since it affects performance and running time. The user should use any available information which gives a good initial approximation. If side constraints exist, the user must be sure the initial values of the design variables do not violate the side constraints. This program will automatically handle an initial design point which is infeasible with respect to the $G(X) < 0$ constraints. However, if the initial point does not violate these constraints, convergence will likely be more rapid.

C. PROBLEM ENTRY

Problem entry is accomplished by editing the main program directly. As an example, consider the following simple NLP with two design variables, and three constraint functions.

Minimize $F(\underline{X}) = X_1^2 + 3\, X_1 X_2 + 2\, X_2^2 - X_1 - X_2 + 1$

subject to ;

$$X_1 + X_2 - 3 \leq 0$$

$$\frac{1}{X_1} + \frac{1}{X_2} - 2 \leq 0$$

$$X_1^2 + X_1 - X_2 - 2 \leq 0$$

$$X_i \geq 0.1$$

With the MSCOP loaded into memory and listed on the CRT, modifications are made on the program lines as follows to input this example :

Line 100

Just after the word "data", three integers are added, separated by a comma. The first number is NDV which is the number of design variables, the second is NIQC which is the number of inequality constraints, and the third is IPRT which is print control number (0 ; only final results, 1 ; given data and final results, 2 ; given data and iterative suboptimal results)

for example :

```
100 data 2,3,2
```

Lines 201-220

Each line here corresponds to a separate design variable, beginning with X(1) and continuing in order to input X(NDV). On each line, three values are separated by commas. After the word "data", these values are the initial values of the design variable, the lower bound on the variable and the upper bound on the variable. If no bound is to be specified, the entry is filled by "no".

For the sample problem, the input is :

```
201 data 3.,0.1,no
202 data 3.,0.1,no
```

Lines 400 - 450

These lines are available for defining the objective function. The objective function must be defined in terms of subscripted design variables X(1), X(2), etc.

For the sample problem, the input is :

```
400 fn_f = x(1)**2+x(1)*x(2)+2.*x(2)**2-x(1)-x(2)+1.
```

Lines 500-650

These lines are available for defining the inequality constraint functions, which must be expressed using the format :

601 if i = k then fn_g = $G_i(x) - b_i$

For the sample problem, the input is :

```
00601 if i = 1 then fn_g = x(1)+x(2)-3.
00602 if i = 2 then fn_g = 1./x(1)+1./x(2)-2.
00603 if i = 3 then fn_g = x(1)**2+x(1)-x(2)-2.
```

If there are many constant values in the constraint functions, the user may input data for these functions on lines 501-600 in order to simplify their statements.

IV. EXAMPLE PROBLEMS

A. DESIGN OF CANTILEVERED BEAM

1. Uniform Cantilevered Beam

Assume a cantilevered beam as shown in Figure 4.1 must be designed. The objective is to find the minimum

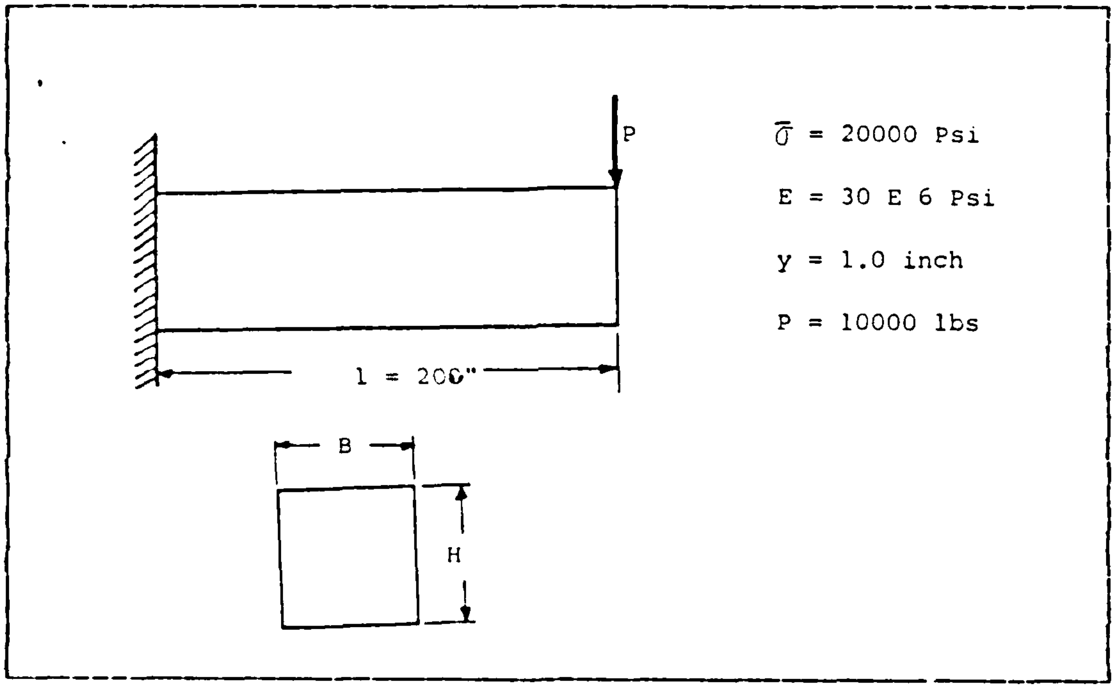


Figure 4.1 Design of a Uniform Cantilevered Beam.

volume of material which will support the load P.

The design variables are the width B and height H in the beam. The design task is as follows : Find B and H to

minimize volume $V = B\ H\ l$ (4.1)

we wish to design the beam subject to limit on bending stress, shear stress, deflection and geometric conditions. The bending stress in the beam must not exceed 20,000 psi.

$$\sigma_b = \frac{M\ c}{I} = \frac{6\ P\ l}{B\ H^2} \leq 20,000 \tag{4.2}$$

The shear stress must not exceed 10,000 psi.

$$\sigma_h = \frac{3\ P}{2\ A} = \frac{3\ P}{2\ B\ H} \leq 10,000 \tag{4.3}$$

and the deflection under the load must not exceed 1 inch.

$$\delta = \frac{P\ l^3}{3\ E\ I} = \frac{4\ P\ l^3}{E\ B\ H^3} \leq 1.0 \tag{4.4}$$

Additionally, geometric limits are imposed on the beam size.

$$0.5 \leq B \leq 5.0 \tag{4.5}$$

$$1.0 \leq H \leq 20.0 \tag{4.6}$$

$$H/b \leq 10. \tag{4.7}$$

Now we can input this problem to MSCOP.

Input NDV, NIQC, IPRT

```
 00100 data 2,4,2
```

Initial starting points

```
 00210 data 3.5,0.5,5.0
 00220 data 16.0,1.0,20.0
```

Evaluation of objective

```
 00400 fn_f = tl*x(1)*x(2)
```

Evaluation of constraints

```
00500 tl = 200.
00501 be = 30.e+6
00502 bp = 10000.
00503 if i = 1 then fn_g = 6.*bp*tl/(20000.*b*h** 2)-1.
00503 if i = 2 then fn_g = 3.*bp/(10000.*2.*b*h)-1.
00503 if i = 3 then fn_g = 4.*bp*tl**3/(be*b*h**3)-1.
00503 if i = 4 then fn_g = h/b-10.
```

TABLE I

The Solution of a Uniform Cantilevered Beam

objective ; 6664.0

design variable ;

X(1) = 1.852

X(2) = 17.99

constraint ;

g(1) = 0.000902

g(2) = -0.9549

g(3) = -0.0109

g(4) = -0.0286

As a result of this problem are in Table 4.1.

2. Variable Cantilevered Beam

The cantilevered beam shown in Figure 4.2 is to be designel for minimum material volume. The design variables are the width b and height h at each of 5 segments. We wish to design the beam subject to limits on stress(calculated at left end of each segment), deflection under the load, and the geometric requirement that the height of any segment does not exceed 20 times the width.

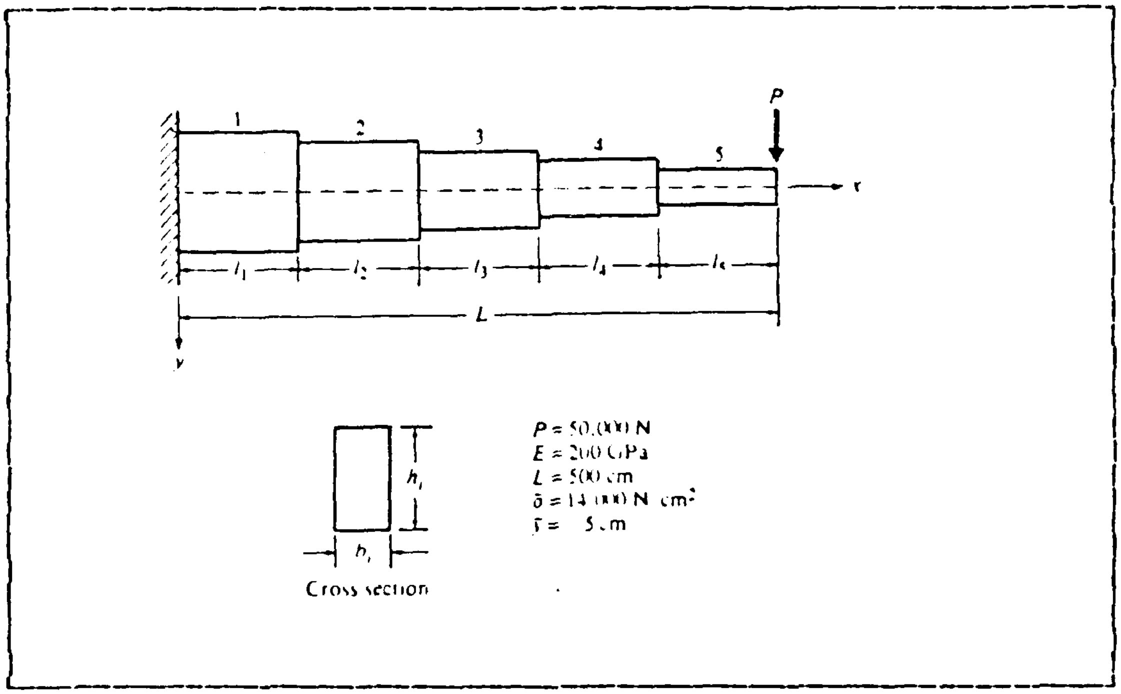


Figure 4.2 Design of a Variable Cantilevered Beam.

The deflection y at the right end of segment i is calculated by the following recursion formulas :

$$y_o = y'_o = 0 \tag{4.8}$$

$$y' = \frac{P\,l_i}{E\,I_i}\left[L + \frac{l_i}{2} + \sum_{j=1}^{i} l_i \right] + y'_{i-1} \tag{4.9}$$

$$y = \frac{P\,l_i^2}{2\,E\,I_i}\left[L - \sum_{j=1}^{i} l_i + \frac{2\,l_i}{3} \right] + y'_{i-1}\,l_i + y_{i-1} \tag{4.10}$$

where the deflection y is defined as positive downward, y' is the derivative of y with respect to the X, and l_i is the length of of segment i. Young's modulus E is the same for all segments, and the moment of inertia for segment i is

$$I_i = \frac{b_i h_i^3}{12} \tag{4.11}$$

The bending moment at the left end of segment i is calculated as

$$M_i = P \left[L + l_i - \sum_{j=1}^{i} l_i \right] \tag{4.12}$$

and the corresponding maximum bending stress is

$$\sigma_i = \frac{M_i h_i}{2 I_i} \tag{4.13}$$

The design task is now defined as

$$\text{Minimize} \quad : \quad V = \sum_{i=1}^{N} b_i h_i l_i \tag{4.14}$$

Subject to : (4.15)

$$\frac{\sigma_i}{\bar{\sigma}} - 1 \leq 0 \qquad i = 1,\ldots,N \tag{4.16}$$

$$\frac{y_N}{\bar{y}} - 1 \leq 0 \tag{4.17}$$

$$h_i - 20\, b_i \leq 0 \qquad i = 1,\ldots,N \tag{4.18}$$

$$b_i \geq 1.0 \qquad h_i \geq 5.0 \qquad i = 1,\ldots,N \qquad (4.19)$$

where $\bar{\sigma}$ is the allowable bending stress and $\bar{y}$ is the allowable displacement. This is a design problem in 10 variables. There are 6 nonlinear constraints defined by Eq(4.16) and Eq(4.17), and 5 linear constraints defined by Eq(4.18), and 10 side constraints on the design variables defined by Eq(4.19).

Now we can input this problem to MSCOP.

Input NDV, NIQC, IPRT

```
00100 data 10,11,2
```

Initial starting points

```
00210 data 5.,1.,no
00220 data 5.,1.,no
00230 data 5.,1.,no
00240 data 5.,1.,no
00250 data 5.,1.,no
00260 data 40.,5.,no
00270 data 40.,5.,no
00280 data 40.,5.,no
00290 data 40.,5.,no
00300 data 40.,5.,no
```

Evaluation of objective

```
00400 fn_f = 100.   * (  x(1)*x(6)   + x(2)*x(7)   + x(3)*x(8)
x(4)*x(9) + x(5)*x(10) )
```

Evaluation of constraints.

```
00490 def fn_g(x,i)
00498 dim bm(10),bi(10),sigi(10),ypb(10),yb(10)
00500 pcb = 50000.
00501 be = 200.e+5
00502 tl = 200.
00503 sigb = 14000.
00504 ytb = .5
00505 sl = 40.
00506 for m = 1 to 5
00507     bm(m) = pcb*(tl+sl-m*sl)
00508 next m
00509 for m = 1 to 5
00510     km = m+5
00511 bi(m) = x(m)*x(km)**3/12.
00512     sigi(m) = bm(m)*x(km)/(2.*bi(m))
00513 next m
00514 yzo = 0.
00515 ypzo = 0.
00516 for m = 1 to 5
```

```
00517     ypb(m) = (pcb*sl*(tl+sl/2.-m*sl))/(be*bi(m))-ypzo
00518     dumm = pcb*sl**2*(tl-m*sl+2.*sl/3.)
00519     yb(m) = dumm/(2.*be*bi(m))+ypzo*sl+yzo
00520     ypzo = ypb(m)
00521     yzo = yb(m)
00522 next m
00550 rem constraint function.
00560 if i = 1 then fn_g = sigi(1)/sigb-1.
00570 if i = 2 then fn_g = sigi(2)/sigb-1.
00580 if i = 3 then fn_g = sigi(3)/sigb-1.
00590 if i = 4 then fn_g = sigi(4)/sigb-1.
00600 if i = 5 then fn_g = sigi(5)/sigb-1.
00610 if i = 6 then fn_g = yb(5)/ybb-1.
00620 if i = 7 then fn_g = x(6)-20.*x(1)
00630 if i = 8 then fn_g = x(7)-20.*x(2)
00632 if i = 9 then fn_g = x(8)-20.*x(3)
00634 if i = 10 then fn_g = x(9)-20.*x(4)
00636 if i = 11 then fn_g = x(10)-20.*x(5)
```

TABLE II

The Solution of a Variable Cantilevered Beam

objective ; 62133.35

design variables	constraints
X(1) = 2.994	G(1) = -0.00219
X(2) = 2.782	G(2) = -0.00415
X(3) = 2.528	G(3) = -0.00508
X(4) = 2.208	G(4) = -0.00406
X(5) = 1.761	G(5) = -0.0177
X(6) = 59.88	G(6) = -0.4401
X(7) = 55.62	G(7) = -0.0101
X(8) = 50.56	G(8) = -0.0231
X(9) = 44.14	G(9) = 0.0000
X(10) = 35.19	G(10) = -0.0248
	G(11) = -0.0278

D. SIMPLE TRUSS

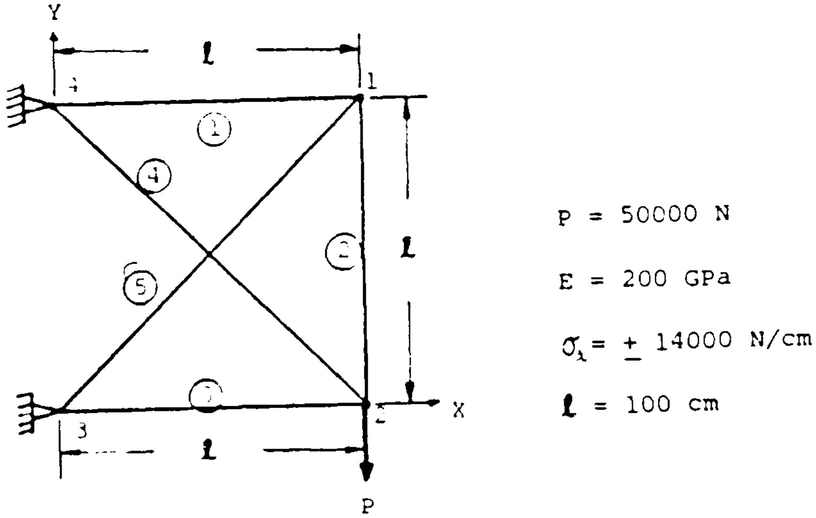


Figure 4.3 Design of a 5-Bar Truss.

A simple truss with 5 members as shown in Figure 4.3 is designed for the minimum volume. The design variables are the sectional areas of the members. The constraints are formed for the stresses of the members not to exceed the given allowable stress. The lower bound for each design variable is also considered. The stresses are obtained by the displacement method of the finite element analysis. The equation to be solved is given by

$$\underline{\underline{K}} \cdot \underline{u} = \underline{P} \tag{4.20}$$

where $\underline{\underline{K}}$ is the stiffness matrix, $\underline{u}$ is the displacement vector and $\underline{P}$ is the load vector as follows :

$$\underline{U} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \end{bmatrix} \qquad \underline{P} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -5000 \end{bmatrix} \tag{4.21}$$

$$\underline{\underline{K}} = E \begin{bmatrix} \frac{A_1}{\ell} + \frac{A_5}{\sqrt{2}\,\ell} & \frac{A_5}{\sqrt{2}\,\ell} & 0 & 0 \\ \frac{A_5}{\sqrt{2}\,\ell} & \frac{A_2}{\ell} + \frac{A_5}{\sqrt{2}\,\ell} & 0 & -\frac{A_2}{\ell} \\ 0 & 0 & \frac{A_3}{\ell} + \frac{A_4}{\sqrt{2}\,\ell} & -\frac{A_4}{\ell} \\ 0 & -\frac{A_2}{\ell} & -\frac{A_4}{\ell} & \frac{A_2}{\ell} + \frac{A_4}{\sqrt{2}\,\ell} \end{bmatrix} \tag{4.22}$$

From Eq. (4.20) the displacements are solved by

$$\underline{U} = \underline{\underline{K}}^{-1} \cdot \underline{P} \tag{4.23}$$

Having displacements at all nodes, we can calculate the stress for each element.

$$\sigma_i = E \cdot \varepsilon = \frac{E \cdot \Delta l_i}{l_i} \tag{4.24}$$

where

$$\Delta l_1 = \sqrt{(l_1 + u_1)^2 + v_1^2} - l_1$$

$$\Delta l_2 = \sqrt{(l_2 + v_1 - v_2)^2 + (u_1 - u_2)^2} - l_2$$

$$\Delta l_3 = \sqrt{(l_3 + u_2)^2 + v_2^2} - l_3 \tag{4.25}$$

$$\Delta l_4 = \sqrt{(l_3 + u_2)^2 + (l_2 - v_2)^2} - l_4$$

$$\Delta l_5 = \sqrt{(l_3 + u_1)^2 + (l_2 + v_1)^2} - l_5$$

The design problem is given by

$$\text{minimize} \quad V = \sum_{i=1}^{5} A_i \, l_i \tag{4.26}$$

Subject to

$$G_i = \frac{|\sigma_i|}{\sigma_a} - 1.0 \leq 0 \quad i = 1,\ldots,5 \tag{4.27}$$

$$A_i \geq 0.1 \qquad i = 1,\ldots,5 \tag{4.28}$$

The MSCOP input for this problem is given as follows :

Input NDV, NIQC, IPRT

```
 00100 data 5,5,2
```

Initial starting point

```
 00200 data 3.,.1,no
 00202 data 3.,.1,no
 00204 data 3.,.1,no
 00206 data 3.,.1,no
 00208 data 3.,.1,no
```

Evaluation of objective

```
 00400 fn f = 100 * ( x(1) + x(2)  + x(3)  + sqr(2.)*x(4)  +
sqr(2.)*x(5))
```

Evaluation of constraints

```
 0500 dim vv(5)
 0501 te = 2.e+7
 0502 tl = 100.
 0503 sigb = 14000.
```

```
0504  cs = 2.*sqr(2.)
0505 ct = te/tl
0506 k11 =  (x(1)+x(5)/cs)*ct
0507 k12 = x(5)*ct/cs
0508 k21 = k12
0509 k22 =  (x(2)+x(5)/cs)*ct
0510 k24 = -x(2)*ct
0511 k33 =  (x(3)+x(4)/cs)*ct
0512 k34 = -x(4)*ct/cs
0513 k42 = k24
0514 k43 = k34
0515 k44 =  (x(2)+x(4)/cs)*ct
0516 dk1 = -k11/k12
0517 dk2 = -(k12+k22*dk1)/k24
0518 dk3 = -k33*dk2/k34
0519 pp = -50000.
0520 vv(1)  = pp/(k42*dk1+k43*dk3+k44*dk2)
0521 vv(2)  = dk1*vv(1)
0522  vv(3)  = dk3*vv(1)
0523 vv(4)  = dk2*vv(1)
0590 dl1 = sqr((tl+vv(1))**2+vv(2)**2)-tl
0592 dl2 = sqr((tl+vv(2)-vv(4))**2+(vv(1)-vv(3))**2)-tl
0594 dl3 = sqr((tl+vv(3))**2+vv(4)**2)-tl
0595 hl = sqr(2.)*tl
0596 dl4 = sqr((hl+vv(3))**2+(hl-vv(4))**2)-hl
0598 dl5 = sqr((hl+vv(1))**2+(hl+vv(2))**2)-hl
0600 rem constraint.
0602 if i = 1 then fn_g = te*dl1/(tl*sigb)-1.
0604 if i = 2 then fn_g = te*dl2/(tl*sigb)-1.
0606 if i = 3 then fn_g = te*dl3/(tl*sigb)-1.
0608 if i = 3 then fn_g = te*dl4/(hl*sigb)-1.
0610 if i = 5 then fn_g = te*dl5/(hl*sigb)-1.
0650 fnend
```

TABLE III

The Sclution of a 5-Bar Truss

objective ; 108.52

design variables	constraint
X(1) = 0.1	G(1) = -1.9988
X(2) = 0.1	G(2) = -2.0030
X(3) = 3.514	G(3) = -0.0030
X(4) = 4.948	G(4) = -0.1203
X(5) = 0.1	G(5) = -1.8797

V. SUMMARY AND CONCLUSION

Numerical optimization is a powerful technique for those confronted with practical engineering design problems. It is also a useful tool for obtaining reasonable solutions to the classical engineering design problems. Since many engineers are now using microcomputers for solving design problems, the development of microcomputer software which can be easily used is needed.

In this thesis, an algorithm for constrained optimization problems is programmed in standard BASIC language (WBASIC version 2.0) on an IBM 3033. The users can easily convert this to other microcomputers.

MSCOP (Microcomputer Software for Constrained Optimization Problems) employs the method of feasible directions and specific modifications of a one-dimensional search for constrained optimization. MSCOP has been validated by tests on three constrained optimization problems. Its performance is good and could be made better through refinement of the algorithm.

Since microcomputers are available with reasonable memory size and computational speed, their capabilities will continue to improve as more engineering software becomes available. MSCOP is considered to be a first step toward more widespread use of optimization techniques on microcomputers.

APPENDIX A

MSCOP PROGRAM LISTING

```
0010 option base 1
0020 dim x(21),x0(21),gcv(51),ngcv(51),df(21),dg(51,21)
0021 dim thta(21),wrky(51,51)
0030 dim a(51,21),b(51,51),p(21),y(21),s(21),u(51),c(51)
0040 dim iwrk(51),jwrk(51),wrk1(51),wrk2(51),wrk3(51)
0050 dim wrku(51),wrkl(51),lowb(21),uprb(21),lo$(6),up$(6)
0060 rem input data
0070 gosub 10000
0080 rem input number of design variables and constraints.
0090 read ndv,nigc,iprt
0100 data 2,4,2
0110 for i = 1 to ndv
0115 rem  input initial value of design variables
0120    read x(i)
0125    x0(i) = x(i)
0130    if nigc = 0 then 160
0135    read lo$,up$
0140    if lo$ = 'no' then lowb(i) = bnlo else lowb(i) =
        value(lo$)
0150    if up$ = 'no' then uprb(i) = bnup else uprb(i) =
        value(up$)
0160 next i
0200 data 3.5,0.5,10.
0210 data 16.,1.0,20.
0360 rem evalute the objective-function
0370 obj = fn_f(x)
0375 itri = 1
0380 rem objective function.
0390 def fn_f(x)
0400    fn_f = 200.*x(1)*x(2)
0410 fnend
0420 rem evaluate the constraints
0430 for i = 1 to nigc
0440    gcv(i) = fn_g(x,i)
0450 next i
0480 rem constraint functions
0490 def fn_g(x,i)
0500 tl = 200.
0510 be = 30.e+6
0520 bp = 10000.
0530 if i = 1 then fn_g = (6.*bp*tl)/
                          (20000.*x(1)*x(2)**2)-1.
0540 if i = 2 then fn_g = (3.*bp)/(20000.*x(1)*x(2))-1.
0550 if i = 3 then fn_g = (4.*bp*tl**3)/
                          (be*x(1)*x(2)**3)-1.
0560 if i = 4 then fn_g = x(2)/(10.*x(1))-1.
0650 fnend
0700 rem initial counting number input
0710 ical = 1
0720 if ical > 3 then stop
0730 rem call the optimization code.
0740 gosub 2000
0750 rem print results.
0760 rem
0770 rem re-counting number input.
0780 ical = ical+1
0785 if ical = 3 then 850
0790 rem 10% reduce the design variables.
0800  for i = 1 to ndv
```

```
0810      x(i) = 0.9*x(i)
0820      x0(i) = x(i)
0830    next i
0840  goto 720
0850  rem 10% increase design variables.
0860  for i = 1 to ndv
0870      x(i) = 1.1*x(i)
0880      x0(i) = x(i)
0890  next i
0900  goto 720
2000  rem calculate the obj. constraint fcn.
2001  obj = fn_f(x)
2002  for i = 1 to nigc
2003      gcv(i) = fn_g(x,i)
2004  next i
2008  itrq = 1
2010  itrq = itrq+1
2020  rem calculate the number of active and violate
          constraints.
2030  gosub 3500
2040  rem calculate the gradient of objective and
          active or violated constraints.
2050  gosub 3800
2060  if navc = 0 then 2190
2070  gosub 3900
2080  rem calculate the push-off factors
2090  gosub 4000
2100  rem making the matrix c
2110  rem normalized the df(i)
2120  gosub 4100
2130  rem normalized the DG(i)
2140  gosub 4200
2150  if nvc > 0 then gosub 4400 else gosub 4600
2160  rem calaulate the usable-feasible direction s(i)
2170  gosub 5000
2180  goto 2230
2190  rem normalize the df(i)
2200  for i = 1 to ndv
2210      s(i) = -(df(i))
2220  next i
2230  rem normalize the s(i)
2240  gosub 5700
2250  rem one-dimensional search
2260  if nvc = 0 then gosub 6000 else gosub 9000
2270  rem update x for alph
2280  gosub 7000
2290  gosub 7100
2300  rem calculate new point value.
2310  nobj = fn_f(x)
2320  rem convergence test
2330  gosub 6780
2340  if walp <= accx and delf <= dabf then 2470
2350      itri = itri+1
2360      if itri > mxit then print 'check the problem'
2370      obj = nobj
2380  for i = 1 to ndv
2390      x0(i) = x(i)
2400  next i
2410  for i = 1 to nicc
2420      gcv(i) = fn_g(x,i)
2430  next i
2440  if iprt = 2 then 2460
2450      gosub 9200
2460  goto 2010
2470  rem print final results
2480  print '***** final results ***** '
2490      gosub 9200
2500  return
3000  rem initialize the integer working array
```

```
3005 for i = 1 to nigm
3010    iwrk(i) = 0
3015 next i
3020 return
3050 rem initialize the integer working array
3055 for i = 1 to nigm
3060    jwrk(i) = 0
3065 next i
3070 return
3100 rem initialize the one-dimension working array
3105 for i = 1 to nigm
3110    wrk1(i) = 0.
3115 next i
3120 return
3150 rem initialize the one-dimension working array
3155 for i = 1 to nigm
3160    wrk2(i) = 0.
3165 next i
3170 return
3200 rem initialize the one-dimension working array
3205 for i = 1 to nigc
3210    wrk3(i) = gcv(i)
3215 next i
3220 return
3250 rem initialize the two-dimension working array
3255 for i = 1 to nigm
3260    for j = 1 to ndvm
3265       wrky(i,j) = 0.
3270    next j
3275 next i
3280 return
3300 rem initialize the derivative of objective DF(i)
3305 for i = 1 to ndvm
3310    df(i) = 0.
3315 next i
3320 return
3350 rem initialize the a(i,j),p(i),y(i),c(i)
3353 for i = 1 to ndvm
3356    p(i) = 0.
3359    y(i) = 0.
3362    for j = 1 to nigm
3365       a(j,i) = 0.
3368    next j
3371 next i
3374 for j = 1 to nigm
3377    c(j) = 0.
3380 next j
3383 return
3400 rem initialize the derivative of constraints DG(i,j)
3405 for i = 1 to nigm
3410    for j = 1 to ndvm
3415       dg(i,j) = 0.
3420    next j
3425 next i
3430 return
3450 rem initialize the b(i,j)
3455 for i = 1 to nigm
3460    for j = 1 to nigm
3465       b(i,j) = 0.
3470    next j
3475 next i
3480 return
3500 rem Calculate the number of active and violate
        constraints.
3502 gosub 3000
3504 gosub 3100
3510 nac = 0
3520 nvc = 0
3530 for i = 1 to nigc
```

```
3540    if gcv(i) >= vcc then 3580
3550    if gcv(i) < acc then 3590
3560       nac = nac+1
3570    goto 3590
3580    nvc = nvc+1
3590 next i
3610 navc = nac+nvc
3620 if navc = 0 then 3790
3630    ii = 1
3640    jj = 1
3650    for i = 1 tc niqc
3660       if gcv(i) >= vcc then 3720
3670       if gcv(i) < acc then 3750
3680          iwrk(nvc+ii) = i
3690          wrk1(nvc+ii) = gcv(i)
3700          ii = ii+1
3710       goto 3750
3720       iwrk(jj) = i
3730       wrk1(jj) = gcv(i)
3740       jj = jj+1
3750    next i
3790 return
3800 rem calculate the gradient of f(x)
3805 gosub 3300
3810 for i = 1 to ndv
3815 dxi = fdm*abs(x(i))
3820    if dxi <= mfds then dxi = mfds
3825       x(i) = x(i)+dxi
3830    dobj = fn_f(x)
3835    df(i) = (dobj-obj)/dxi
3840    x(i) = x0(i)
3850 next i
3860 return
3900 rem calculate the DG(i,j)
3901 gosub 3400
3910 for i = 1 to ndv
3915    dxi = fdm*x(i)
3920    if dxi < mfds then dxi = mfds
3925    x(i) = x(i)+dxi
3930    for j = 1 tc navc
3935       k = iwrk(j)
3940       dcon = fn_g(x,k)
3945       dg(j,i) = (dcon-wrk1(j))/dxi
3950    next j
3955    x(i) = x0(i)
3960 next i
3966 return
4000 rem calcilate the push-off factor
4010 for i = 1 to navc
4020    thta(i) = tht0*(1.-wrk1(i)/acc)**2
4030    if thta(i) > thtm then thta(i) = thtm
4040 next i
4090 return
4100 rem normalize the DF(i)
4102 gosub 3200
4105 fsq = 0.
4110 for i = 1 to ndv
4115    fsq = fsq+df(i)**2
4120 next i
4125 fsq = sqr(fsq)
4127 if fsq = 0. then fsq = zro
4130 for i = 1 to ndv
4135    wrk3(i) = (1./fsq)*df(i)
4140 next i
4145 return
4200 rem normalize the DG(i)
4202 gosub 3250
4205 for i = 1 to navc
4210    gsq = 0.
```

```
4215      for j = 1 to ndv
4220         gsq = gsq+dg(i,j)**2
4225      next j
4230      gsq = sqr(gsq)
4232      if gsq = 0. then gsq = zro
4235      for j = 1 to ndv
4240         wrky(i,j) = (1./gsq)*dg(i,j)
4245      next j
4250   next i
4255   return
4400   rem exist the violate constraints
4405   gosub 3350
4410   for i = 1 to navc
4420      for j = 1 to ndv
4430         a(i,j) = wrky(i,j)
4440      next j
4450      a(i,ndv+1) = thta(i)
4460   next i
4470   for i = 1 to ndv
4480      p(i) = -wrk3(i)
4490   next i
4500   p(ndv+1) = phid
4510   for i = 1 to navc
4520      yy = 0
4530      for j = 1 to ndv+1
4540         xx = a(i,j)*p(j)
4550         yy = yy+xx
4560      next j
4570      c(i) = (-1.)*yy
4580   next i
4586   ndb = navc
4590   return
4600   rem only exist active constraints
4605   gosub 3350
4610   for i = 1 to navc
4620      for j = 1 to ndv
4630         a(i,j) = wrky(i,j)
4640      next j
4650      a(i,ndv+1) = thta(i)
4660   next i
4670   for j = 1 to ndv
4680      a(navc+1,j) = wrk3(j)
4690   next j
4700   a(navc+1,ndv+1) = 1.
4710   p(ndv+1) = 1.
4720   for i = 1 to navc+1
4730      cc = a(i,ndv+1)*p(ndv+1)
4740      c(i) = (-1.)*cc
4750   next i
4760   ndb = navc+1
4770   return
5000   rem  calculate the usable-feasible direction
5002   gosub 3000
5005   gosub 3250
5010   gosub 3450
5040   for i = 1 to ndb
5050      for j = 1 to ndv+1
5060         wrky(j,i) = a(i,j)
5070      next j
5080   next i
5090   for i = 1 to ndb
5100      for j = 1 to ndb
5110         ff = 0.
5120         for k = 1 to ndv+1
5130            tf = a(i,k)*wrky(k,j)
5140            ff = ff+tf
5150         next k
5160         b(i,j) = (-1.)*ff
5170      next j
```

```
5180 next i
5190 iter = 0
5200 nmax = 5*ndb
5210 rem begin iteration
5220 iter = iter+1
5230 cbmx = 0.
5240 ichk = 0
5250 for i = 1 to ndb
5260    ci = c(i)
5270    bii = b(i,i)
5280    if bii = 0. then 5340
5290    if ci > 0. then 5340
5300       cb = ci/bii
5310    if cb <= cbmx then 5340
5320    ichk = i
5330    cbmx = cb
5340 next i
5350 if cbmx < zro or iter > nmax then 5550
5360 if ichk = 0 then 5550
5370    jj = iwrk(ichk)
5380 if jj = 0 then iwrk(ichk) = ichk else iwrk(ichk) = 0
5385    if b(ichk,ichk) = 0. then b(ichk,ichk) = zro
5400       bb = 1./b(ichk,ichk)
5405       if bb = 0. then bb = zro
5410       for i = 1 to ndb
5420          b(ichk,i) = bb*b(ichk,i)
5430       next i
5440       c(ichk) = cbmx
5460       for i = 1 to ndb
5470          if i = ichk then 5530
5480          bbi = b(i,ichk)
5490          b(i,ichk) = 0.
5500          for j = 1 to ndb
5505          if j = ichk then 5520
5510             b(i,j) = b(i,j)-bbi*b(ichk,j)
5520          next j
5525       c(i) = c(i)-bbi*cbmx
5530       next i
5540 goto 5220
5550 ner = 0
5560 for i = 1 to ndb
5565    u(i) = 0.
5570    j = iwrk(i)
5580 if j > 0 then u(i) = c(j)
5600 next i
5610 for i = 1 to ndb
5620    ff = 0.
5630    for j = 1 to ndb
5640       ff = ff+wrky(i,j)*u(j)
5650    next j
5660    y(i) = p(i)-ff
5670    s(i) = y(i)
5680 next i
5690 return
5700 rem normalized the s(i)
5710 ssq = 0.
5720 for i = 1 to ndv
5730    ssq = ssq+s(i)**2
5740 next i
5750 ssq = sqr(ssq)
5755 if fslp = 0. then fslp = zro
5760 for i = 1 to ndv
5770    s(i) = (1./ssq)*s(i)
5780 next i
5820 return
6000 rem one-dimensional search for initial feasible point.
6005 rem calculate for slope of f(x)
6010 fslp = 0.
6015 for i = 1 to ndv
```

```
6020    fslp = fslp+df(i)*s(i)
6025 next i
6035 rem idenfy the initial point.
6040 alow = 0.
6045 flow = obj
6050 for i = 1 to niqc
6055    wrk1(i) = gcv(i)
6060 next i
6065 rem find a1st ; the 1st mid-point.
6067 if fslp = 0. then fslp = zro
6070 a1st = aboj*flow/abs(fslp)
6075 for i = 1 to ndv
6076    if s(i) = 0. then s(i) = zro
6080    walp = alpx*x(i)/abs(s(i))
6085    if walp > a1st then 6095
6090       a1st = walp
6095 next i
6100 rem update x for a1st.
6105 alph = a1st
6110 gosub 7000
6115 gosub 7100
6120 rem calculate the f1st and wrk1(i)
6125 f1st = fn_f(x)
6130 for i = 1 to niqc
6135    wrk1(i) = fn_g(x,i)
6140 next i
6145 rem check the feasibility.
6150 ncv1 = 0
6155 for i = 1 to niqc
6160    if wrk1(i) < vcc then 6170
6165       ncv1 = ncv1+1
6170 next i
6175 if ncv1 = 0 then 6200
6180 a1st = 0.5*a1st
6185 goto 6105
6190 rem find a2nd ; the 2nd mid-point.
6195 rem 2-points quadratic fit interpolation
         for minimum f(alpa).
6200 a0 = flow
6205 a1 = fslp
6210 a2 = (f1st-a1*a1st-a0)/(a1st**2)
6215 if a2 <= 0. then a2 = zro
6220    a2nd = -a1/(2.*a2)
6225 rem 2-points linear interpolation for g(alpa)=0.
6230 for i = 1 to niqc
6235    a0 = wrk1(i)
6237    if a1st = 0. then a1st = zro
6240    a1 = (wrk1(i)-a0)/a1st
6245    if a1 <= 0. then a1 = zro
6250    walp = -a0/a1
6252 if walp <= 0. then walp = 1000.
6255    if walp >= a2nd then 6265
6260       a2nd = walp
6265 next i
6270 rem update x for a2nd.
6275 alph = a2nd
6280 gosub 7000
6285 gosub 7100
6290 rem calculate f2nd and wrk2(i)
6295 f2nd = fn_f(x)
6300 for i = 1 to niqc
6305    wrk2(i) = fn_g(x,i)
6310 next i
6315 rem find final point appr by using
         3-points quadratic fit.
6320 f1 = flow
6321 alp1 = alow
6325 f2 = f1st
6326 alp2 = a1st
```

```
6330 f3 = f2nd
6331 alp3 = a2nd
6335 rem 3-points quadratic fit interpolation.
6340 gosub 6600
6342 if a2 = 0. then a2 = zro
6345 a3rd = -a1/(2.*a2)
6347 if a3rd <= 0. then a3rd = 1000.
6350 for i = 1 to niqc
6355    f1 = wrk1(i)
6360    f2 = wrk1(i)
6365    f3 = wrk2(i)
6370    gosub 6600
6375    gosub 6630
6376 if alps > a3rd then 6380
6377         a3rd = alps
6380 next i
6385 rem update x for aupr
6390 alph = a3rd
6395 gosub 7000
6400 gosub 7100
6405 rem calculate the fupr and wrku(i)
6410 fupr = fn_f(x)
6415 for i = 1 to niqc
6420    wrku(i) = fn_g(x,i)
6425 next i
6430 rem find 4th new point.
6435 f1 = f1st
6440 f2 = f2nd
6445 f3 = f3rd
6450 alp1 = a1st
6455 alp2 = a2nd
6460 alp3 = a3rd
6465 rem 3-points quadratic fit.
6470 gosub 6600
6475    if a2 = 0. then a2 = zro
6480    aupr = -a1/(2.*a2)
6485 for i = 1 to niqc
6490    f1 = wrk1(i)
6495    f2 = wrk2(i)
6500    f3 = wrk3(i)
6505    alp1 = a1st
6510 alp2 = a2nd
6515    alp3 = a3rd
6520    gosub 6600
6525    gosub 6630
6530    if alps > aupr then 6540
6535    aupr = alps
6540 next i
6545 rem update x for aupr
6550 alph = aupr
6555 gosub 7000
6560 gosub 7100
6565 rem evaluate fupr and wrku(i)
6567 fupr = fn_f(x)
6569 for i = 1 to niqc
6571      wrku(i) = fn_g(x,i)
6573 next i
6575 rem find optimum alpa for not violating constraints.
6577 gosub 14300
6579 return
6600 rem 3-points quadratic fit.
6603 if alp1 = alp2 or alp2 = alp3 or alp1 = alp3
     then return
6605 a2 = ((f3-f1)/(alp3-alp1)-
          (f2-f1)/(alp2-alp1))/(alp3-alp2)
6610 a1 = (f2-f1)/(alp2-alp1)-a2*(alp1+alp2)
6615 a0 = f1-a1*alp1-a2*alp1**2
6620 return
6630 rem zero of polynomial for g(alpa)
```

```
6635 dd = a1**2-4.*a2*a0
6640 if dd < 0. then 6715
6642 if a2 <= 0. then a2 = zro
6645 if a2 = 0. then a2 = zro
6650    alpb = (-a1+sqr(dd))/(2.*a2)
6655    alpc = (-a1-sqr(dd))/(2.*a2)
6660    if alpb <= 0 and alpc <= 0. then 6715
6665    if alpb >= 0. and alpc >= 0. then 6695
6670    if alpb >= 0. and alpc < 0. then 6685
6675       alps = alpc
6680 goto 6720
6685       alps = alpb
6690    goto 6720
6695    if alpb >= alpc then 6710
6700       alps = alpb
6705    goto 6720
6710       alps = alpc
6712    goto 6720
6715 alps = 1000.
6720 return
6780 rem update aboj and alpx
6790 delf = abs(obj-nobj)
6795 diff = abs(delf/obj)
6800 aboj = (aboj+diff)/2.
6815 walp = 0.
6816 welx = 0.
6820 for i = 1 to ndv
6830    delx = abs(x0(i)-x(i))
6850    difx = abs(delx/x0(i))
6855    if delx >= welx then welx = delx
6860    if difx <= walp then 6880
6870       walp = difx
6880 next i
6890 alpx = (alpx+walp)/2.
6910 daff = accf*abs(obj)
6990 return
7000 rem update the x(i)
7010 for i = 1 to ndv
7020    x(i) = x0(i)+alph*s(i)
7030 next i
7040 return
7100 rem check the side-constraints.
7110 for i = 1 to ndv
7120    if x(i) < lowb(i) then x(i) = lowb(i)
7130    if x(i) > uprb(i) then x(i) = uprb(i)
7140 next i
7150 return
8000 rem estimate the alpa
8010 fstr = flow
8020 alpa = alow
8030 nvc1 = 0
8040 for i = 1 to nigc
8050    if wrk1(i) < vcc then 8070
8060    nvc1 = nvc1+1
8070 next i
8080 if nvc1 > 0 then 8120
8090 if f1st > fstr then 8120
8100    alpa = a1st
8110    fstr = f1st
8120 nvc1 = 0
8130 for i = 1 to nigc
8140    if wrk2(i) < vcc then 8160
8150    nvc1 = nvc1+1
8160 next i
8170 if nvc1 > 0 then 8210
8180 if f2nd > fstr then 8210
8190    alpa = a2nd
8200    fstr = f2nd
8210 nvc1 = 0
```

```
8220 for  i = 1 to nigc
8230     if wrk3(i)  < vcc then 8250
8240     nvc1 = nvc1+1
8250 next i
8260 if nvc1 > 0 then 8300
8270 if f3rd > fstr then 8300
8280     alpa = a3rd
8290     fstr = f3rd
8300 nvc1 = 0
8310 for i = 1 to nicc
8320   if wrku(i)  < vcc then 8340
8330     nvc1 = nvc1+1
8340 next i
8350 if nvc1 > 0 then 8390
8360 if fupr > fstr then 8390
8370     alpa = aupr
8380     fstr = fupr
8390 alph = alpa
8400 return
9000 rem one-dimensional search for initial
         infeasible point.
9002 ii = 1
9004 gcvm = wrk1(1)
9006 for i = 1 to navc
9008 if wrk1(i)  <=  gcvm then 9014
9010     ii = i
9012     gcvm = wrk1(i)
9014 next i
9016 rem calculate the slope of badly violation.
9018 gslp = 0.
9020 for i = 1 to ndv
9022     gslp = gslp+dg(ii,i)*s(i)
9024 next i
9026 rem calculate the alph.
9027 if gslp = 0. then gslp = zro
9028 alph = -gcvm/gslp
9030 rem update X for alph.
9032 gosub 7000
9034 gosub 7100
9036 rem evalute the objective and constraint.
9038 obj = fn_f(x)
9040 for i = 1 to nigc
9042     gcv(i)  = fn_g(x,i)
9044 next i
9046 rem calculate the NVC.
9048 gosub 3500
9050 if nvc = 0 then return
9052 rem update initial value.
9054 for i = 1 to ndv
9056     x0(i)  = x(i)
9058 next i
9060 rem calculate df(i),dg(i,j) and push-off factor.
9062 gosub 3800
9064 gosub 3900
9066 gosub 4000
9068 rem normalize the df(i),dg(i,j)
9070 gosub 4100
9072 gosub 4200
9074 rem find the search direction.
9076 gosub 4400
9078 gosub 5000
9080 goto 9000
9200 rem print the results
9205 print ''
9210 print '*********** data ***********'
9215 print ''
9220 print 'The number of design variables      = ',ndv
9225 print 'The number of inequality constraints = ',nigc
9230 print ''
```

```
9235 print 'The objective value = ',obj
9240 print ''
9245 print '***** design variables *****'
9250 for i = 1 to ndv
9255     print 'x(';i;') = ',x(i)
9260 next i
9265 print ''
9270 print 'the number of active constraints = ';nac
9275 print ''
9280 print 'the number of vionate constraints = ';nvc
9285 print ''
9290 print '**** constraint value ****'
9295 print ''
9300 for i = 1 to nicc
9305     print 'g(';i;') = ';gcv(i)
9310 next i
9315 return
9500 rem default number
9510 mxit = 50    ! maximum iteration number
9520 fdm = .01    ! finite difference step
9530 mfds = .001 ! maximum absolute finite difference ste
9540 vcc = .004  ! violated constraint criteria (thickness)
9550 acc = -.1   ! active constraints criteria (thickness)
9560 tht0 = 1.   ! push-off factor multiplier (theta zero)
9570 thtm = 50.  ! maximum value of push-off factor
9580 phid = 100000. ! weighting-factor used in direction
                   when infeasible
9590 accf = .001   ! absolute convergence criteria
9600 accx = 0.001  ! absolute convergence criteria.
9610 zro = .0001   ! defined zero
9620 espl = .005  ! used to prevent division by zero
9630 bnlo = -1.e+70 ! the value of low boundary
9640 bnup = 1.e+70  ! the value of upper boundary
9650 dalp = .01  ! step size of alpa in one-dimensional
                 search
9660 abcj = 0.1   ! step size for reduce objective
9670 alpx = .1    ! reduce the design variable factor
9680 ndvm = 21  ! the number of maximum design variable
9690 nigm = 51   ! the number of maximum inequality
                 constraints
9700 return
9800 end
```

LIST OF REFERENCES

1. Noriaki Yoshida, (H.I.T in Japan) " Optimum Stiffener Design to High Stress Region of Beam Column Criteria", 37th JSCE Conferance, NO. 10, 1982

2. L. Madsen and G. N. Vanderplaats, "COPES - A FORTRAN Control Program for Engineering Synthesis ", Naval Postgraduate School Report, NPS 69-81-003, 1982

3. G. N. Vanderplaats, "ADS - A FORTRAN Program for Automated Design Synthesis. Version 1.0", Naval Postgraduate School, May 1984.

4. G. N. Vanderplaats, "Structural Optimization - Past, Present, Future", AIAA Journal Vol. 20, No 7. July 1982.

5. G. N. Vanderplaats, Numerical Optimization Method for Engineering Design with Applications, Mc Graw - Hill, May 1984.

6. G. N. Vanderplaats, "Feasible Direction Method ", Naval Postgraduate School, July 1978.

7. G. N. Vanderplaats, "A Robust Feasible Directions Algorithm for Design Synthesis ", AIAA/ASME/ASCE/AHS, 24th Structures, Structural Dynamics and Materials Conferance, Lake Tahoe, Nevada, May 2-4, 1983.

8. Zoutendijk, G, Methods of Feasible Direction, Elsevier, Amsterdam, 1960.

9. R. S. Johnson, Optimization Design of Mechanical Elements, 2nd edition, Willy-Interscience Publication John Willy and Sons, 1980.

BIBLIOGRAPHY

Harney M. Wagner, Principles of Operations Research with Applications to Managinal decisuon, Presentice-Hall Englewood cliff, NJ. 1969

Byron S. Gottfried, Programming with BASIC Including Microcomputer BASIC, 2nd Edition Schaum Outline Series in Computers, McGraw-Hill Book Company, 1980

J. W Graham, J. W. Welch, K. I. McPhee, Waterloo BASIC / A Structural Programming Approach / Primer and Referance Manual, WATCOM Publications, Waterloo Ontario N2J 4C3, 1983

H. Falk, " Software Tool for Mechanical Engineers", J. Mech. Engr., Vol. 195 No. 8, August 1983.

M. S. Bazaraa and C. M. Shetty, Nonlinear Programming : Theory and Algorithms, John Willy & Sons, 1979

Leonard Spunt, Optimum Structural Design, Prentice - Hall, Inc. 1971.

E. Whitman Wright, Structural Design by Computer, Van Nostrand Reinhold Company LTD, 1976.

N. G. R. Iyengar and S. K. Gupta, Programming Method in Structural Design, John Willy & Sons, 1980.

R. L. Booth, Development of a Microcomputer - Based Engineering Design Optimization Software Package, Naval Postgraduate School Engineer Thesis, Dec. 1983.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	Department Chairman, Code 54 Dept. of Operations Research Naval Postgraduate School Monterey, California 93943	1
4.	Professor G. N. Vanderplaats, Dept. of Mechanical Engineering Univ. of California at Santa Babara California 93106	3
5.	Professor R. K. Wood ,Code 55 Wd Dept. of Operations Research Naval Postgraduate School Monterey, California 93943	2
6.	Dr. H. Miura M.S. 237-11 NASA Ames Research Center Moffett Field, California 94035	1
7.	Professor Noriyaki Yoshida Dept. of Civil Engineering 419 Teine Maeda, Nishi-ku Sapporo, Japan 061-24	1
8.	Professor Yong S. Shin, Code 69 Sg Dept. of Mechanical Engineering Naval Postgraduate School Monterey, California 93943	1
9.	Professor T. Sarpkaya, Code 69 Sl Dept. of Mechanical Engineering Naval Postgraduate School Monterey, California 93943	1
10.	Professor Robert R. Nunn, Code 69 Nn Dept. of Mechanical Engineering Naval Postgraduate School Monterey, California 93943	1
11.	Professor M. D. Kelleher, Code 69 Kk Dept. of Mechanical Engineering Naval Postgraduate School Monterey, California 93943	1
12.	Professor Gilles Cantine, Code 69 Ci Dept. of Mechanical Engineering Naval Postgraduate School Monterey, California 93943	1

13. Professor Seong Hwan, Cho 1
Dept. Mechanical Engineering
PO Box 77. Kong Neung Dong
Do Bong Ku, Seoul,Korea 130-09

14. Dong Soo, Kim 5
585-15 Hyoja 1 dong
Chuncheon city KangWondo.
Seoul Korea 200

15. Personnel Management Office 2
Army Headquarters
Seoul, Korea 140-01

16. Office of the Defense Attache 1
Embassy of the Republic of Korea
2320 Massachusetts Avenue, Northwest
Washington, D.C. 20008

17. Library Officer 2
Korea Military Academy
Seoul, Korea 130-09